|  |  |  |  |
| --- | --- | --- | --- |
| SImplesoft:**ManualTECNICO**    **DIRECCIÓN WEB: http:simplesoft.hosting.cs.umss.edu.bo**  **E-MAIL DEL PROPONENTE**  simple.soft@hotmail.com |  |  | **materia**  **taller de ingenieria de software**  **CONSULTOR TIS**  **lic. david escalera fernandez**  **integrantes**  **Augusto Lafuente Reyes**  **Virgilio Pacheco Choque**  **ronal Choque Copa**  **Henrri Yapura Quispe**  **semestre**  **1/2017** |

**DESCRIPCIÓN DEL PROBLEMA**

Debido a la necesidad de un ambiente para las labores académicas dentro de los predios de la universidad Mayor de San Simón, sea optado por una organización manual de asignación de horarios específicos vinculando un ambiente, una fecha, y un usuario al cual se le esta asignando dicho ambiente, todo este proceso es registrado en un cuaderno por parte de un administrador el cual al verificar ambientes existentes procede a la designación a un nuevo usuario caso contrario si existe reservas el usuario tendrá que buscar otras opciones, cronológicamente el guía ante cualquier reservas es el calendario académico que provee la facultad, este método manual dificulta mucho la consulta ya que se tiene que buscar el lugar físico donde se encuentra, no tiene mucha información adicional en tiempo real, por este motivo es necesario optar por un sistema de reserva de ambientes.

**JUSTIFICACIÓN**

Este sistema permitirá automatizar la reserva de reserva de ambientes en los predios de la Universidad Mayor de San Simón de esta manera mejorar y hacer mas eficiente el manejo de asignación de ambientes mediante reservas.

**ESPECIFICACIONES TÉCNICAS**

Sistema Operativo: Windows 7, 8.1, 10 o GNU/Linux 16.04 o versiones superiores

Base de datos: MySQL

Leguaje de Programación: PHP 5.6, HTML5, CSS3, JavaScript, Jquery

Navegador Web: Google Chrome, Firefox, Opera

IDE: Sublime Text 3

ANÁLISIS DE REQUERIMIENTOS

Para el desarrollo del sistema se consideró la siguiente lista de requerimientos.

* Formulario de reserva.
* Control de calendario académico.
* Reserva de ambientes.
* Modificación de reserva.
* Búsquedas de fechas libres.
* Servidor web provisto por el departamento de informática y Sistemas.
* Confidencialidad y seguridad.
* Administración de usuarios y roles.

Consultas a reservas existentes.

**DOCUMENTACIÓN DE REFERENCIA**

Trello: [https://trello.com/](https://trello.com/%20)

GIT: <http://git-scm.com/documentation/>

FullCalendar: <https://fullcalendar.io/>

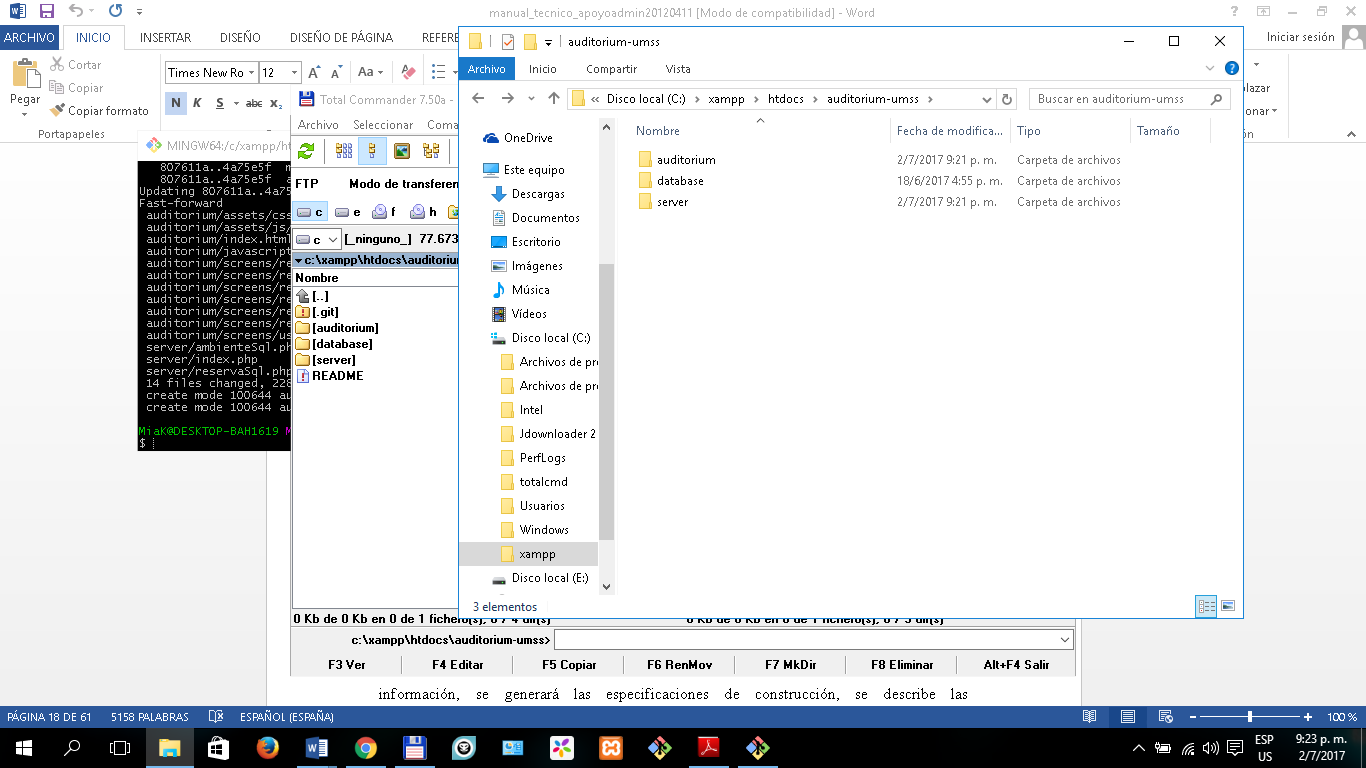
QTip: <http://qtip2.com/guides>

Magnific-Popup: <http://dimsemenov.com/plugins/magnific-popup/documentation.html>

MomentJS: <http://momentjs.com/docs/>

**ESTRUCTURA DEL PROYECTO**

A continuación se presenta la siguiente estructura de directorios y archivos utilizados para la realización del proyecto.

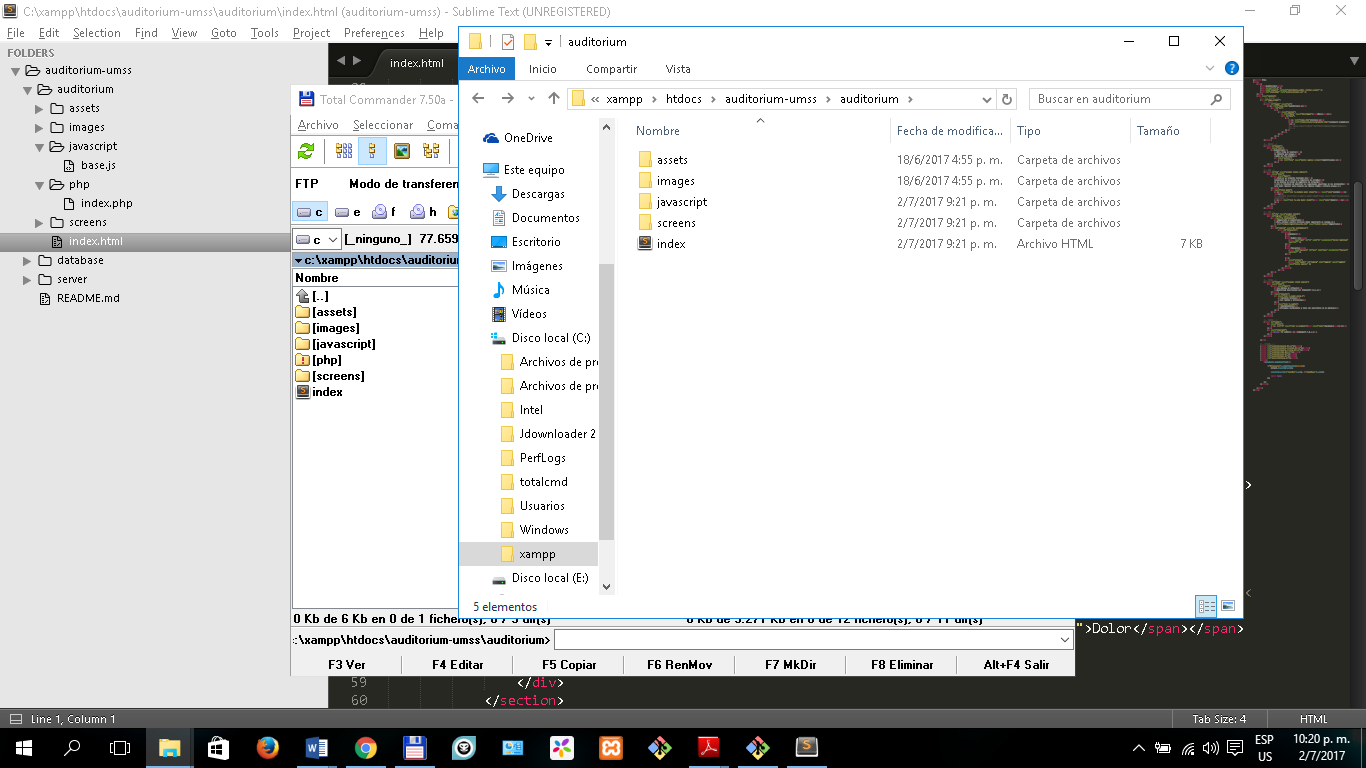


**Figura 1: estructura principal.**

**DIRECTORIO AUDITORIUM**

Este directorio contiene un subdirectorio en donde se encuentra todos los archivos necesarios para la interacción con el usuario desde una página web.

Este directorio contiene subdirectorios los cuales se muestran a continuación.



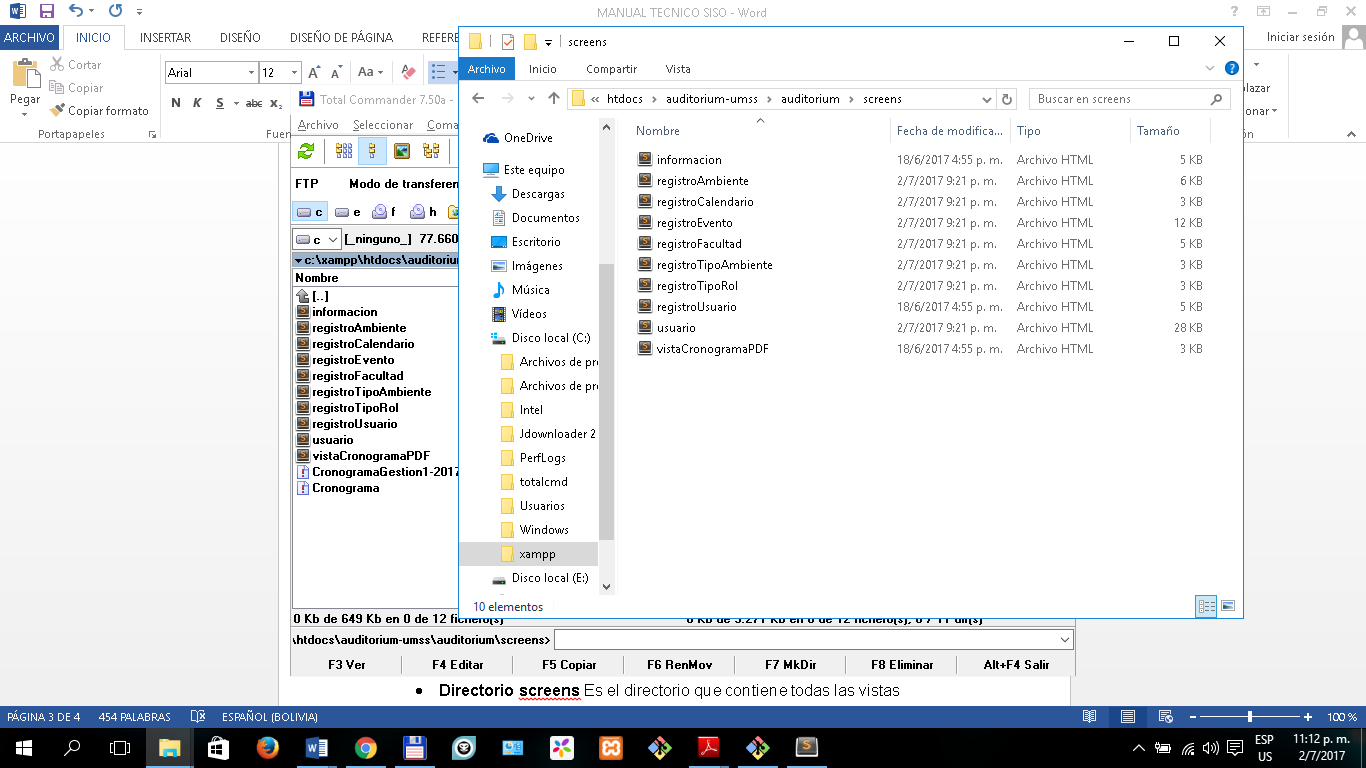
**Figura 2: subdirectorios del directorio auditórium.**

* **Directorio assets**  Es el directorio q contiene código de librerías externas

JQuery, FullCalendar, MomentJs, MagnificPopup, QTip, hojas d estilo y código JavaScript d las librerías externas y el témplate del proyecto.

* **Directorio images** Es el directorio que contiene todas las imágenes utilizadas en las diferentes vistas.
* **Directorio javascript** Es el directorio donde se encuentra el código necesario para la conexión de vistas de interfaz de usuario con el servidor el cual está en otro directorio.
* **Directorio screens** Es el directorio que contiene todas las vistas desde donde interactúa el usuario

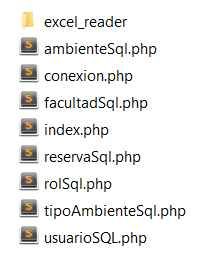
.



**DIRECTORIO DATABASE**

Es el directorio que contiene el script sql el cual contiene toda la configuración necesaria para inicial el sistema completamente limpio.

**DIRECTORIO SERVER**



Este directorio contiene la conexión a la Base de Datos, las diferentes consultas q se necesitan realizar a la BD.

También contiene el acceso principal para las consultas HTTP q se realizan al servidor desde el Cliente (index.php)

La Carpeta “e*xcel\_reader*” contiene una librería q usamos para leer un archivo .xls o .xlsx y con los datos del calendario de la gestión y asi poder cargar esa información a la BD.

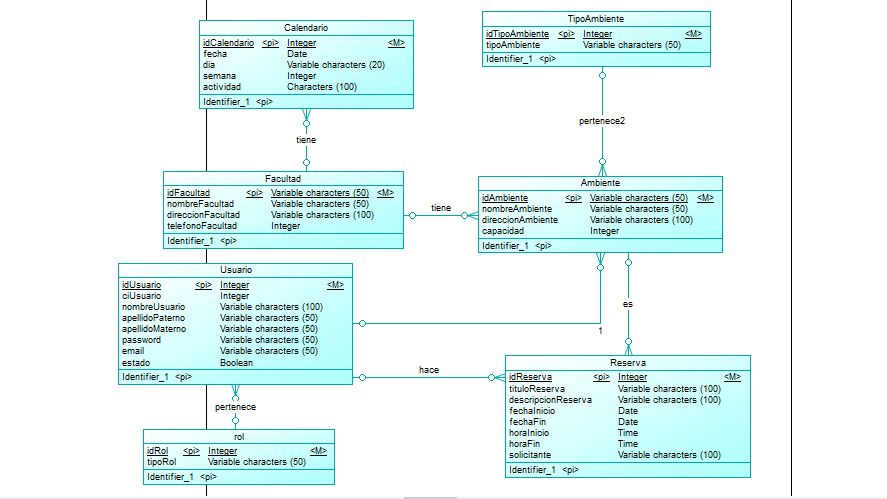
**FRAMEWORKS**

En el proyecto se utilizaron los siguientes frameworks:

* JQUERY es una [biblioteca](https://es.wikipedia.org/wiki/Biblioteca_(inform%C3%A1tica)) [multiplataforma](https://es.wikipedia.org/wiki/Multiplataforma) de [JavaScript](https://es.wikipedia.org/wiki/JavaScript), creada inicialmente por [John Resig](https://es.wikipedia.org/wiki/John_Resig), que permite simplificar la manera de interactuar con los documentos [HTML](https://es.wikipedia.org/wiki/HTML), manipular el árbol [DOM](https://es.wikipedia.org/wiki/Document_Object_Model), manejar eventos, desarrollar animaciones y agregar interacción con la técnica [AJAX](https://es.wikipedia.org/wiki/AJAX) a páginas web.
* FULLCALENDAR es un plugin de jQuery escrito por Adam Shaw que presenta datos en forma de calendario o agenda. Puede cargar los eventos directo por AJAX (aunque también por otros mecanismos).
* MOMENT.JS es una librería para Javascript que nos facilitan la vida al trabajar con fechas en javascript, y es que nos permite parsear, validar, formatear y manipular las fechas más fácilmente y lo mejor de todo que soporta múltiples idiomas o los famosos Locale.
* MAGNIFICPOPUP es una variación de una manera común de mostrar imágenes o texto en una ventana modal. El efecto se suele llamar una "caja de luz", el nombre de una de las bibliotecas originales de JavaScript que lo utilizaron. Hay muchas bibliotecas JavaScript diferentes que puede utilizar para lograr este mismo efecto. Magnifica permite el uso de varios tipos de contenido en sus modales: imágenes, en línea (html), iframe y AJAX.
* QTIP es un plugin jQuery para mostrar información extra sobre links, lo que se suele denominar como tooltips. Ha sido construido a partir de cero para ser fácil de usar, sin embargo, es rico en características, Qtip proporciona una infinidad de opciones, como esquinas redondedas, tamaño, color del texto, fondo, etc.
* PHPEXCEL es un framework creada en PHP que nos permitirá exportar y leer diferentes formatos de Excel, como .xls (Excel 2007), .xlsx (OfficeOpenXML), CSV, PDF, HTML, entre otros.

**DISEÑO DE BASE DE DATOS**

Para el modelado de la base de datos se utilizó el programa Power Designer, se hizo un modelo entidad relación el cual se muestra a continuación:



**Figura 3: modelo entidad relación.**

**PUNTOS CLAVE**

En todo el proyecto existen dos puntos clave y d estos depende la fortaleza y el buen funcionamiento del sistema completos:

* base.js, en este archivo encontraremos las llamadas *$.ajax*, estas llamadas son el puente principal de comunicación desde el cliente con el servidor. De esta manera enviamos consultas HTTP con los datos necesarios (filtros, datos a registrar, datos a actualizar, etc) para realizar algún tipo de procesamiento en el servidor, para luego recibir y procesar/mostrar la respuesta.
* index.php, este es el punto de entrada de toda consulta HTTP q se envié al servidor. Un vez llega la consulta se verifica el tipo, GET o POST, para luego redirigir a una función específica encargada de procesar ese tipo de consulta. Normalmente las consultas POST es usaran para realizar cambios en el sistema (registros, actualizaciones, reservas, etc), pero en el caso de las consultas GET las usamos para obtener alguna información del servidor y enviarlo d regreso al cliente ya sea para mostrarlo o procesarlo d alguna manera, pero d ninguna manera se debe realizar ningún cambio en estado del sistema con este tipo de consultas.

**CLIENTE**

En el lado del cliente el funcionamiento es simple e intuitivo, cada una de las pantallas tiene un código JavaScript q se encarga de establecer el funcionamiento de cada uno de los componentes de esa pantalla (normalmente compuesto de formularios).

El caso de los registros (usuario, ambiente, reserva, etc), se obtiene todos los datos de los campos haciendo uso de la librería jQuery y se las envía a alguna d las funciones existentes en *base.js*, dependiendo de lo q se necesite realizar.

Cuando hablamos del Login al sistema, enviamos los datos ingresados en el formulario de conexión al servidor, realizamos las validaciones necesarias para verificar la existencia y validez de los datos. Una vez verificada la identidad se envía una respuesta con la información del usuario conectado (id, nombre, apellidos), una vez en el cliente se almacenan los datos de la respuesta en un cache temporal del explorador para ser usados luego desde el cliente:

* sessionStorage.idUsuario
* sessionStorage.nombreUsuario
* sessionStorage.apellido1
* sessionStorage.apellido2

Una vez se cierre el cliente toda esta información se eliminara del explorador automáticamente.

Toda consulta enviada al servidor contiene un objeto con la siguiente configuración:



Los puntos más importantes d esta configuración son:

* type, el tipo de método HTTP
* data, el objeto enviado al servidor con los datos necesarios para ser procesados.
* success, evento callback q se lanza cuando la comunicación con el servidor fue exitosa, el parámetro ‘obj’ es el objeto q el servidor responde una vez terminado el proceso en ese lado.

**SERVIDOR**

En el caso del servidor la lógica se basa en su totalidad del lenguaje PHP. En este lado del sistema, nuestro punto de entrada, index.php, hace de proxi para todas las consultas HTTP q se envían al servidor y se los redirecciona a las respectivas funciones dependiendo del atributo ‘accion’ q se envía desde el cliente como parte de consulta.

Además del atributo ‘acción’ es posible q se envíen otros datos para ser procesados en el servidor. En la mayoría de los casos, los datos extras nos llega en el atributo ‘parametros’ el cual es una lista.

El Objeto ‘data’ enviado desde el cliente lo encontramos en las variables ‘$\_POST’ o ‘$\_GET’ cuando llegan al servidor, una vez sabiendo q tipo de consulta es lo enviamos a una de las funciones específicas para cada uno, runGet o runPost.

En cada una de estas funciones nos encargamos de obtener y evaluar cada uno de los parámetros enviados desde el cliente, evaluando primero el parámetro ‘accion’ para saber cuál es la acción q se debe ejecutar para esta llamada, esto se lo hace con el uso de una estructura *switch*.

Finalmente, una vez procesada la información en el servidor se llena la respuesta en la variable $result, este luego es enviado al cliente y se lo puede acceder en el callback *success* mediante el parametro *obj.*